JASPERREPORTS - REPORT EXPRESSION

Report expressions are a powerful feature of JasperReports, which allows us to display calculated data on a report. Calculated data is the data that is not a static data and is not specifically passed as a report parameter or datasource field. Report expressions are built from combining report parameters, fields, and static data. By default, the Java language is used for writing report expressions. Other scripting languages for report expressions like Groovy scripting language, JavaScript or BeanShell script are supported by JasperReports compilers.

This chapter will explain you how report expressions work assuming that they have been written using the Java language only. In a JRXML report template, there are several elements that define expressions, like the following :

- <variableExpression>
- <initialValueExpression>
- <groupExpression>
- <printWhenExpression>
- <imageExpression>
- <textFieldExpression>

Expression Declaration

Basically, all report expressions are Java expressions that can reference report fields, report variables and report parameters.

Field Reference In Expression

To use a report field reference in an expression, the name of the field must be put between **$F{** and **}** character sequences, as shown below.

```
<textfieldexpression>
   $F{Name}
</textfieldexpression>
```

Following is a piece of code from our existing JRXML file, from chapter Report Designs:

```
<textFieldExpression >
    <![CDATA[$F{country}]]>
</textFieldExpression>
```

Variable Reference In Expression

To reference a variable in an expression, we must put the name of the variable between **$V{** and **}** like in the example below:

```
<textfieldexpression>
   "Total height : " + $V{SumOfHeight} + " ft."
</textfieldexpression>
```

Parameter Reference In Expression

To reference a parameter in an expression, the name of the parameter should be put between **$P{** and **}** like in the following example:

```
<textfieldexpression>
   "ReportTitle : " + $P{Title}
</textfieldexpression>
```

Following is a piece of code from our existing JRXML file, which demonstrates the referencing of parameter in an expression. (JRXML from chapter Report Designs):

```
<textField isBlankWhenNull="true" bookmarkLevel="1">
    <reportElement x="0" y="10" width="515" height="30"/>
    <textElement textAlignment="Center">
        <font size="22"/>
    </textElement>
    <textFieldExpression >
       <![CDATA[$P{ReportTitle}]]>
    </textFieldExpression>
    <anchorNameExpression>
       <![CDATA["Title"]]>
    </anchorNameExpression>
</textField>
<textField isBlankWhenNull="true">
    <reportElement  x="0" y="40" width="515" height="20"/>
    <textElement textAlignment="Center">
        <font size="10"/>
    </textElement>
    <textFieldExpression >
       <![CDATA[$P{Author}]]>
    </textFieldExpression>
</textField>
```

As you can see above, the parameter, field, and variable references are in fact real Java objects. Knowing their class from the parameter, field or variable declaration made in the report template, we can even call methods on those object references in the expressions.

Following example shows how to extract and display the first letter from java.lang.String report field "Name":

```
<textFieldExpression>
    $F{Name}.substring(0, 1)
</textFieldExpression>
```

Resource Bundle Reference In Expression

To reference a resource in an expression, the *key* should be put between **$R{** and **}** like in the following example:

```
<textfieldexpression>
   $R{report.title}
</textfieldexpression>
```

Based on the runtime-supplied locale and the *report.title* key, the resource bundle associated with the report template is loaded. Hence the title of report is displayed by extracting the String value from the resource bundle. More on internationalization can be found in the chapter Internationalization.

Calculator

Calculator is an entity in JasperReports, which evaluates expressions and increments variables or datasets at report-filling time. During compile process, the information is produced and stored in the compile report by the compiler. This information is used during the report-filling time to build an instance of the net.sf.jasperreports.engine.fill.JRCalculator class.

Java source file is generated and compiled by Java-based report compilers on the fly. This generated class is a subclass of the JRCalculator, and the bytecode produced by compiling it is stored inside the JasperReport object. This bytcode is loaded at the report filling time and the resulting class is instantiated to obtain the calculator object needed for expression evaluation.

Conditional Expressions

Jasper Reports doesn't support if-else statements when defining variable expressions. Instead you can use the ternary operators **{cond} ? {statement 1} : {statement 2}**. You can nest this operator inside a Java expression to obtain the desired output based on multiple conditions.

Example of conditional Expression in Report

Let's modify existing report template (Chapter Report designs) and add a conditional expression for field country. The revised report template (jasper_report_template.jrxml) is as follows. Save it to C:\tools\jasperreports-5.0.1\test directory:

```
<?xml version="1.0"?>
<!DOCTYPE jasperReport PUBLIC
"//JasperReports//DTD Report Design//EN"
"http://jasperreports.sourceforge.net/dtds/jasperreport.dtd">

<jasperReport xmlns="http://jasperreports.sourceforge.net/jasperreports"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://jasperreports.sourceforge.net/jasperreports
http://jasperreports.sourceforge.net/xsd/jasperreport.xsd" name="jasper_report_template"
pageWidth="595" pageHeight="842" columnWidth="515" leftMargin="40" rightMargin="40"
topMargin="50" bottomMargin="50">
<parameter name="ReportTitle" />
<parameter name="Author" />
   <queryString>
    <![CDATA[]]>
   </queryString>
   <field name="country" >
      <fieldDescription><![CDATA[country]]></fieldDescription>
   </field>
   <field name="name" >
      <fieldDescription><![CDATA[name]]></fieldDescription>
   </field>
   <sortField name="country" order="Descending"/>
   <sortField name="name"/>
   <title>
      <band height="70">
         <line>
            <reportElement x="0" y="0" width="515"
            height="1"/>
         </line>
         <textField isBlankWhenNull="true" bookmarkLevel="1">
            <reportElement x="0" y="10" width="515"
            height="30"/>
            <textElement textAlignment="Center">
            <font size="22"/>
            </textElement>
            <textFieldExpression >
            <![CDATA[$P{ReportTitle}]]>
            </textFieldExpression>
            <anchorNameExpression><![CDATA["Title"]]>
            </anchorNameExpression>
            </textField>
            <textField isBlankWhenNull="true">
            <reportElement  x="0" y="40" width="515" height="20"/>
            <textElement textAlignment="Center">
                 <font size="10"/>
            </textElement>
            <textFieldExpression >
            <![CDATA[$P{Author}]]>
            </textFieldExpression>
            </textField>
      </band>
   </title>
   <columnHeader>
      <band height="23">
         <staticText>
            <reportElement mode="Opaque" x="0" y="3"
            width="535" height="15"
            backcolor="#70A9A9" />
            <box>
            <bottomPen lineWidth="1.0"
            lineColor="#CCCCCC" />
            </box>
            <textElement />
            <text><![CDATA[]]>
            </text>
         </staticText>
         <staticText>
            <reportElement x="414" y="3" width="121"
            height="15" />
            <textElement textAlignment="Center"
            verticalAlignment="Middle">
                <font isBold="true" />
            </textElement>
            <text><![CDATA[Country]]></text>
         </staticText>
         <staticText>
            <reportElement x="0" y="3" width="136"
            height="15" />
            <textElement textAlignment="Center"
            verticalAlignment="Middle">
               <font isBold="true" />
            </textElement>
            <text><![CDATA[Name]]></text>
         </staticText>
      </band>
    </columnHeader>
    <detail>
      <band height="16">
         <staticText>
            <reportElement mode="Opaque" x="0" y="0"
            width="535" height="14"
            backcolor="#E5ECF9" />
            <box>
               <bottomPen lineWidth="0.25"
               lineColor="#CCCCCC" />
            </box>
            <textElement />
            <text><![CDATA[]]>
            </text>
         </staticText>
         <textField>
            <reportElement x="414" y="0" width="121"
            height="15" />
            <textElement textAlignment="Center"
            verticalAlignment="Middle">
               <font size="9" />
            </textElement>
            <textFieldExpression >
              <![CDATA[$F{country}.isEmpty() ? "NO COUNTRY" : $F{country}]]>
            </textFieldExpression>
         </textField>
         <textField>
            <reportElement x="0" y="0" width="136"
            height="15" />
            <textElement textAlignment="Center"
            verticalAlignment="Middle" />
            <textFieldExpression >
            <![CDATA[$F{name}]]>
            </textFieldExpression>
         </textField>
      </band>
   </detail>
</jasperReport>
```

The java codes for report filling are as follows. The contents of the file **C:\tools\jasperreports-5.0.1\test\src\com\tutorialspoint\JasperReportFill.java** are as below.

```
package com.tutorialspoint;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.Map;

import net.sf.jasperreports.engine.JRException;
import net.sf.jasperreports.engine.JasperFillManager;
import net.sf.jasperreports.engine.data.JRBeanCollectionDataSource;

public class JasperReportFill {
   @SuppressWarnings("unchecked")
   public static void main(String[] args) {
      String sourceFileName =
      "C://tools/jasperreports-5.0.1/test/jasper_report_template.jasper";

      DataBeanList DataBeanList = new DataBeanList();
      ArrayList<DataBean> dataList = DataBeanList.getDataBeanList();

      JRBeanCollectionDataSource beanColDataSource =
      new JRBeanCollectionDataSource(dataList);

      Map parameters = new HashMap();
      /**
       * Passing ReportTitle and Author as parameters
       */
      parameters.put("ReportTitle", "List of Contacts");
      parameters.put("Author", "Prepared By Manisha");

      try {
         JasperFillManager.fillReportToFile(
         sourceFileName, parameters, beanColDataSource);
      } catch (JRException e) {
         e.printStackTrace();
      }
   }
}
```

The contents of the POJO file **C:\tools\jasperreports-5.0.1\test\src\com\tutorialspoint\DataBean.java** are as below:

```
package com.tutorialspoint;

public class DataBean {
   private String name;
   private String country;

   public String getName() {
      return name;
   }

   public void setName(String name) {
      this.name = name;
   }

   public String getCountry() {
      return country;
   }

   public void setCountry(String country) {
      this.country = country;
   }
}
```

We will add a new record with country field as empty in our Java bean List. The contents of the file

C:\tools\jasperreports-5.0.1\test\src\com\tutorialspoint\DataBeanList.java are as below:

```
package com.tutorialspoint;

import java.util.ArrayList;

public class DataBeanList {
   public ArrayList<DataBean> getDataBeanList() {
      ArrayList<DataBean> dataBeanList = new ArrayList<DataBean>();

      dataBeanList.add(produce("Manisha", "India"));
      dataBeanList.add(produce("Dennis Ritchie", "USA"));
      dataBeanList.add(produce("V.Anand", "India"));
      dataBeanList.add(produce("Shrinath", "California"));
      dataBeanList.add(produce("Tanmay", ""));
      return dataBeanList;
   }

   /**
    * This method returns a DataBean object,
    * with name and country set in it.
    */
   private DataBean produce(String name, String country) {
      DataBean dataBean = new DataBean();
      dataBean.setName(name);
      dataBean.setCountry(country);
      return dataBean;
   }
}
```

Report generation

We will compile and execute the above file using our regular ANT build process. The contents of the file build.xml (saved under directory C:\tools\jasperreports-5.0.1\test) are as below.

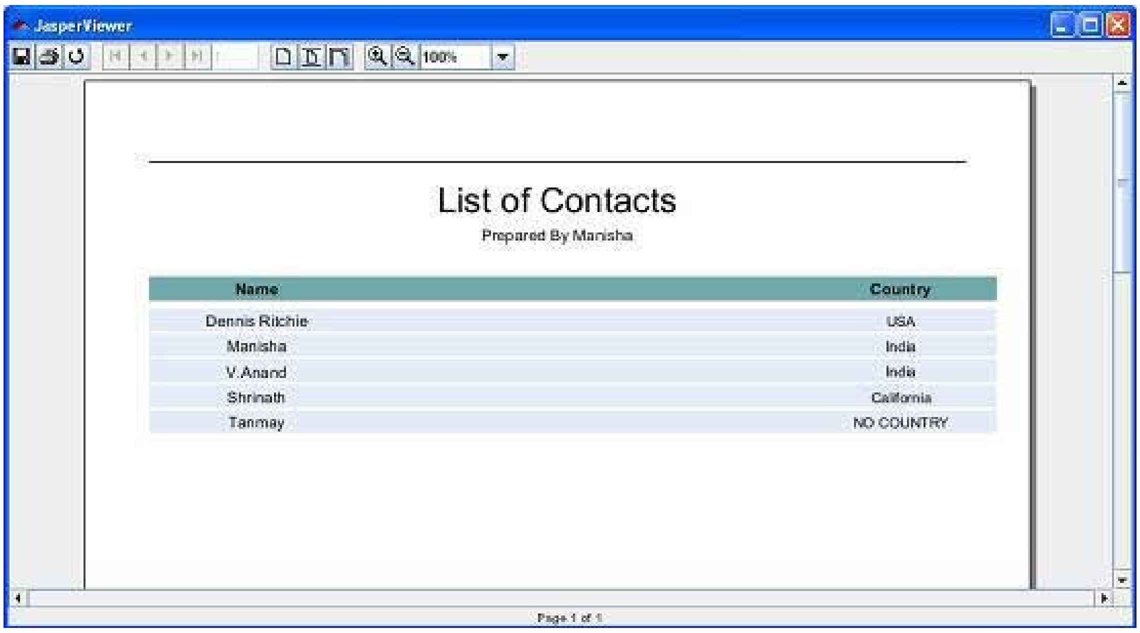
> *The import file - baseBuild.xml is picked from chapter* [Environment Setup] *and should be placed in the same directory as the build.xml.*

```
<?xml version="1.0" encoding="UTF-8"?>
<project name="JasperReportTest" default="viewFillReport" basedir=".">
   <import file="baseBuild.xml" />
   <target name="viewFillReport"
      depends="compile,compilereportdesing,run"
      description="Launches the report viewer to preview
      the report stored in the .JRprint file.">
      <java classname="net.sf.jasperreports.view.JasperViewer"
      fork="true">
         <arg value="-F${file.name}.JRprint" />
         <classpath ref />
      </java>
   </target>
   <target name="compilereportdesing"
      description="Compiles the JXML file and
      produces the .jasper file.">
      <taskdef name="jrc"
      classname="net.sf.jasperreports.ant.JRAntCompileTask">
         <classpath ref />
      </taskdef>
      <jrc destdir=".">
         <src>
         <fileset dir=".">
            <include name="*.jrxml" />
         </fileset>
         </src>
         <classpath ref />
      </jrc>
   </target>
</project>
```

Next, let's open command line window and go to the directory where build.xml is placed. Finally execute the command **ant -Dmain-class=com.tutorialspoint.JasperReportFill** (viewFullReport is the default target) as follows:

```

C:\tools\jasperreports-5.0.1\test>ant -Dmain-class=com.tutorialspoint.JasperReportFill
Buildfile: C:\tools\jasperreports-5.0.1\test\build.xml

clean-sample:
   [delete] Deleting directory C:\tools\jasperreports-5.0.1\test\classes
   [delete] Deleting: C:\tools\jasperreports-5.0.1\test\jasper_report_template.jasper
   [delete] Deleting: C:\tools\jasperreports-5.0.1\test\jasper_report_template.jrprint

compile:
    [mkdir] Created dir: C:\tools\jasperreports-5.0.1\test\classes
    [javac] C:\tools\jasperreports-5.0.1\test\baseBuild.xml:28:
    warning: 'includeantruntime' was not set, defaulting to build.sysclasspath=last;
    set to false for repeatable builds
    [javac] Compiling 3 source files to C:\tools\jasperreports-5.0.1\test\classes

compilereportdesing:
      [jrc] Compiling 1 report design files.
      [jrc] log4j:WARN No appenders could be found for logger
      (net.sf.jasperreports.engine.xml.JRXmlDigesterFactory).
      [jrc] log4j:WARN Please initialize the log4j system properly.
      [jrc] log4j:WARN See
      http://logging.apache.org/log4j/1.2/faq.html#noconfig for more info.
      [jrc] File : C:\tools\jasperreports-5.0.1\test\jasper_report_template.jrxml ... OK.

run:
     [echo] Runnin class : com.tutorialspoint.JasperReportFill
     [java] log4j:WARN No appenders could be found for logger
     (net.sf.jasperreports.extensions.ExtensionsEnvironment).
     [java] log4j:WARN Please initialize the log4j system properly.

viewFillReport:
     [java] log4j:WARN No appenders could be found for logger
     (net.sf.jasperreports.extensions.ExtensionsEnvironment).
     [java] log4j:WARN Please initialize the log4j system properly.

BUILD SUCCESSFUL
Total time: 5 minutes 5 seconds

C:\tools\jasperreports-5.0.1\test>
```

As a result of above compilation, a JasperViewer window opens up as in the screen below:



List of Contacts

Prepared By Manisha

Name	Country
Dennis Ritchie	USA
Manisha	India
V.Anand	India
Shrinath	California
Tanmay	NO COUNTRY

Here we can see, for the last record we had not passed any data for the field country, "NO COUNTRY" is being printed.